Controle de Inércia para Fuga de Mínimos Locais de Funções Não-Lineares na Otimização por Enxame de Partículas

Tiago Silveira, Humberto César Brandão de Oliveira, Luiz Eduardo da Silva

Departamento de Ciências Exatas - Universidade Federal de Alfenas - MG - Brasil

{tiago,humberto,luizedu}@bcc.unifal-mg.edu.br

Abstract. *This work presents a mechanism to reduce the chances of the optimization process of nonlinear functions stagnating in local minima, using the meta-heuristic Particle Swarm Optimization. This mechanism is a non-monotonic way to control the particle inertia, which is one of the factors responsible for movement during the optimization process. The experimental results were compared to the PSO original model aiming to show the potential to find a better solution in the benchmark functions for complex problems.*

Resumo. *Este trabalho apresenta um mecanismo para reduzir as chances do processo de otimização de funções não-lineares estacionar em mínimos locais, ao utilizar a meta-heurística Otimização por Enxame de Partículas. Tal mecanismo trata-se de uma forma não-monotônica de controlar a inércia da partícula, que é um dos fatores responsáveis pela movimentação dessa durante o processo de otimização. Os resultados experimentais foram comparados com o modelo original da PSO padrão a fim de mostrar o potencial em encontrar uma melhor solução em funções de benchmark, para problemas complexos.*

1. Introdução

A otimização numérica é a tarefa de determinar valores ótimos dentro de um universo de possibilidades ($x \in \Re^n$), onde o grau de otimização é dado por uma função de avaliação ($f : \Re^n \rightarrow \Re$), que pode ser linear ou não-linear [Coelho *et al.* 2004].

As técnicas mais utilizadas para tratar problemas de otimização numérica podem ser divididas em duas classes: os algoritmos específicos para problemas bem definidos, como, por exemplo, as técnicas de programação linear, e os meta-algoritmos de busca, como a otimização por enxame de partículas, que é o objeto de estudo deste trabalho.

O algoritmo de Otimização por Enxame de Partículas (PSO) é um meta-algoritmo de busca motivado no comportamento social. Através da competição e da cooperação entre indivíduos, assim como na natureza, a otimização com base no comportamento social pode trazer diversos benefícios, encontrando soluções boas de forma eficiente, mantendo certa simplicidade no processo de otimização.

Apesar de abordagens diferentes, todos os algoritmos com base na natureza trabalham de formas similares, com a atualização da população de acordo com informações obtidas sobre a adaptabilidade ao ambiente, sendo esperada a busca de uma solução mais promissora [Shi e Eberhart 1999]. Porém, com a complexidade dos

problemas não-lineares, a otimização por tais algoritmos se torna deficiente para encontrar o melhor ponto do espaço, devido ao elevado número de soluções ótimas locais que o problema passa a ter. Um ponto ótimo local geralmente não é a solução desejada. Se por influência de um mínimo local a otimização converge para pontos distantes da solução ótima, o resultado desejado poderá não ser satisfatório. Logo, fazer com que a otimização desses problemas não convirja prematuramente para pontos que venham prejudicar o resultado vem sendo uma tarefa bastante explorada por muitos pesquisadores [Jiao *et al.* 2008, Yang *et al.* 2007, Lvbjerg *et al.* 2001].

Com este intuito, apresentamos neste trabalho um mecanismo para reduzir as chances do processo de otimização de funções não-lineares ficar estacionado em mínimos locais, ao utilizar a meta-heurística Otimização por Enxame de Partículas. Tal mecanismo está relacionado ao controle de forma não-monotônica da inércia da partícula, sendo esta um dos fatores de atualização da velocidade que cada partícula terá durante o processo de otimização. Estando estabilizadas em um mínimo local, as partículas, através do controle da inércia, irão ganhar energia, aumentando suas velocidades e, conseqüentemente, saindo do mínimo local encontrado, podendo assim buscar soluções mais favoráveis. Todo o controle de inércia terá como base a função co-seno, pois fornece esse valor de inércia com o comportamento não-monotônico.

No decorrer deste trabalho mostramos o potencial do mecanismo de controle de inércia da partícula, comparando seus resultados experimentais com a PSO padrão. O restante do trabalho está dividido da seguinte forma: a Seção 2 apresenta uma visão geral do modelo da PSO padrão. Na Seção 3 são citados alguns trabalhos que buscam um melhor desempenho da PSO. A Seção 4 apresenta a forma que este trabalho trata a inércia da partícula na PSO. A Seção 5 descreve as configurações experimentais usadas para encontrar os resultados descritos e discutidos na Seção 6. E, finalmente, a Seção 7 faz uma conclusão do estudo, indicando também as futuras diretrizes do trabalho.

2. Otimização por Enxame de Partículas

O algoritmo de Otimização por Exame de Partículas (PSO, do inglês *Particle Swarm Optimization*) foi introduzido em meados da década de 90 por Kennedy e Eberhart [Kennedy e Eberhart 1995], como uma alternativa ao Algoritmo Genético padrão. Conceitualmente, a PSO é uma técnica de busca estocástica que visa otimizar uma função de objetivo, sendo desenvolvida através da tentativa de simular graficamente a coreografia realizada por pássaros em busca de alimentos. Mais tarde, buscando fundamentos teóricos, foram realizados estudos sobre a maneira como indivíduos em sociedades, de uma forma geral, interagem, trocando informações e revendo seus conceitos em busca de melhores soluções para seus problemas [Kennedy *et al.* 2001].

A PSO tem raízes em 2 principais metodologias componentes. Talvez o mais evidente, são seus laços com a vida artificial em geral. Ela é também relacionada, entretanto, a computação evolucionária, e tem laços com algoritmos genéticos e a programação evolucionária [Kennedy e Eberhart 1995]. Porém, a PSO não utiliza os operadores evolucionários para manipular seus indivíduos, mas uma velocidade é atribuída para cada indivíduo para a movimentação pelo espaço de busca, sofrendo o ajuste de velocidade a cada iteração, de acordo com a sua própria experiência (experiência cognitiva), a experiência das outras partículas (experiência social) do

enxame e sua velocidade atual. A não execução do operador de seleção é uma característica que difere a PSO dos algoritmos genéticos, da programação evolucionária e das estratégias evolucionárias [Eberhart e Shi 1998, Angeline 1998].

Na PSO, cada partícula é dita ser uma possível solução para o problema de otimização. Aqui, todas as partículas são mantidas como membros permanentes da população, sendo que o termo atualizado será a velocidade da partícula. Essa velocidade é a responsável em fazer com que a partícula tente ir para uma região mais promissora, pois trata-se de um vetor que está sempre em busca de uma melhor solução momentaneamente. A PSO torna-se assim o único algoritmo evolucionário que não considera a sobrevivência do indivíduo mais forte [Eberhart e Shi 1998].

A implementação do algoritmo da PSO é dada da seguinte forma: Seja s o tamanho do enxame, n a dimensão do problema e t o instante atual, cada partícula i possui uma posição $x_i(t) \in \Re^n$ no espaço de soluções e uma velocidade $v_i(t) \in \Re^n$ que indica a direção e a magnitude de seu deslocamento. Adicionalmente, cada partícula possui a lembrança $p_i^* \in \Re^n$ da melhor posição individual visitada, e o enxame possui a lembrança da melhor posição visitada por alguma partícula até então ($p_b^* \in \Re^n$). No decorrer do algoritmo, a velocidade de cada partícula é calculada segundo a melhor posição visitada individual p_i^*, a melhor posição visitada do enxame p_b^* e a componente que agrupa sua velocidade anterior, servindo com um termo de *momentum* (inércia). Assim, a atualização da velocidade de cada partícula fica de acordo com a equação (1):

$$v_i(t+1) = wv_i(t) + c_1 r_1 (p_i^*(t) - x_i(t)) + c_2 r_2 (p_b^*(t) - x_i(t)) \quad (1)$$

onde *r1* e *r2* são componentes aleatórias retiradas de uma distribuição uniforme entre 0 e 1, responsáveis por uma busca mais natural, como na natureza, durante o processo de otimização [Kennedy e Eberhart 1995]. Já c_1 e c_2 são os coeficientes de aceleração, que geralmente possuem valores fixos e iguais, responsáveis por controlar a distância que uma partícula irá se mover em apenas uma iteração. O item w é o peso de inércia (termo de *momentum*) que multiplica a velocidade no instante t anterior e faz com que a busca seja mais explorativa no início e mais explotativa no final, para um valor inércia linearmente decrescente, como sugerido por [Kennedy e Eberhart 1995].

Após a atualização da velocidade da partícula, sua posição atual sofre a atualização segundo a equação (2):

$$x_i(t+1) = x_i(t) + v_i(t+1) \quad (2)$$

Como apresentado em [Kennedy e Eberhart 1995], nota-se que a PSO compreende um conceito extremamente simples, necessitando apenas de operadores matemáticos primitivos para sua implementação.

3. Trabalhos Relacionados

A partir da concepção original da PSO, tem se buscado novas adaptações no algoritmo a fim de melhorar seu desempenho. Tal tarefa não se mostra tão simples de ser feita. Como exemplo, para fazer a seleção de parâmetros, no trabalho apresentado em [Shi e Eberhart 1998], fica evidenciado a complexidade para se elaborar tal processo na PSO.

Além das configurações dos parâmetros, vários pesquisadores têm analisado o desempenho da PSO com diferentes configurações, utilizando várias funções para o espaço de busca muito conhecidas na literatura. Por exemplo, nos trabalhos de [Suganthan 1999, Kennedy 1999], são discutidas as configurações de vizinhança das partículas, sendo esta uma característica que afeta o termo social das partículas durante o processo de otimização. O trabalho de [Kennedy 1999] conclui que a vizinhança do tipo estrela (onde a partícula tem o conhecimento da melhor posição encontrada entre todas as partículas) é uma boa opção na busca de melhores soluções na PSO.

Assim como este trabalho, outros estudos também focam no peso de inércia da partícula para buscar um melhor desempenho da PSO. Em [Yang *et al.* 2007] é proposta uma nova estratégia com uma modificação no algoritmo da PSO utilizando um peso de inércia dinâmico, que se ajusta de acordo com a velocidade de evolução e o grau de agregação do enxame. Em [Jiao *et al.* 2008] também é proposta uma PSO que usa um peso de inércia dinâmico, que decresce de acordo com a evolução do enxame.

Outra forma para tentar melhorar o desempenho da PSO é através da hibridização do algoritmo original. Um exemplo seria o trabalho de [Lvbjerg *et al.* 2001]. Com inspiração nos algoritmos genéticos, este trabalha com uma forma de reprodução entre os indivíduos do enxame. Outra idéia testada em [Lvbjerg *et al.* 2001] é o conceito de subpopulações, onde a divisão das partículas em várias subpopulações é feita para tentar manter a diversidade e assim tentar fugir da convergência para mínimos locais.

4. Inércia da Partícula

A partir de agora, focaremos no aspecto principal deste trabalho. O termo de inércia do algoritmo original da PSO será apresentado e discutiremos uma forma em que, alterando o modo original de atuação deste termo, pode-se obter benefícios durante a otimização em relação a PSO padrão, para problemas complexos.

4.1. Termo de Inércia

O peso de inércia tem características que são remanescentes do parâmetro temperatura no *Simulated Annealing* [Eberhart e Shi 1998]. Na PSO, ele foi introduzido a fim de balancear a pesquisa local e global. Um alto peso de inércia facilita uma busca global, enquanto que um baixo peso de inércia facilita a busca local.

Na concepção original da PSO, o processo de otimização tem um bom desempenho com o termo de inércia decaindo linearmente [Shi e Eberhart 1999]. Tendo um valor de peso de inércia alto no início da execução decrescendo linearmente para um valor pequeno durante o processo de otimização, a PSO terá, inicialmente, uma capacidade maior de pesquisa global, enquanto terá uma maior capacidade de busca local aproximando-se do fim do processo. Cientes destas características, já em seu trabalho inicial, [Kennedy e Eberhart 1995] sugerem o peso de inércia monotonicamente decrescente, sendo esta uma subtração linear entre os valores 0,9 e 0,4.

4.2. Controle de Inércia

Uma característica observada em [Shi e Eberhart 1999], é que a PSO pode necessitar da capacidade de pesquisa global no final de uma execução, devido à estagnação em um mínimo local, ao se utilizar um peso de inércia linearmente decrescente. Tal capacidade

é ainda mais necessária caso o problema a ser resolvido seja muito complexo. Assim, na maioria das vezes, a PSO pode ser deficiente para encontrar o ótimo global para esses problemas.

Observando essa necessidade, introduzimos então uma modificação na forma de trabalhar com a inércia durante a execução da PSO. Em nossos testes iniciais, verificamos que ao impor um peso de inércia maior que 1, as partículas começavam a se espalhar pelo espaço de busca. Nesse caso, a velocidade do instante anterior passa a ter maior importância no deslocamento atual da partícula. Isso ocorre porque ao multiplicar a velocidade anterior da partícula por um número maior que 1.0, a nova velocidade resultante será maior (terá maior importância) que a velocidade anterior, fazendo assim com que as partículas ganhem energia (aumentem a velocidade) e se afastem do ponto onde estavam.

Assim, ao invés da inércia *w* decair linearmente ao longo da otimização, ela passaria agora a oscilar durante todo o processo de otimização segundo uma função periódica com comportamento ondulatório. Para isso, com base na função co-seno, a inércia *w* da partícula passou a ter esse comportamento não-monotônico, de acordo com a equação (3):

$$w = \left[\cos\left(\frac{\pi i}{2\,ciclos} \right) \times m \right] + s \qquad (3)$$

onde *i* é a iteração corrente, *ciclos* corresponde ao número de ciclos não-monotônicos utilizados, *m* é um multiplicador, gerado pela diferença do valor máximo de *w* pelo valor mínimo de *w* (informados antes do início do processo), que depois é divida por 2 *((w_max – w_min)/2)*, responsável por calcular a metade da altura da função para os valores de *w* utilizados. O deslocamento da função é calculado por *s*, que o faz com a soma de *m* com o valor mínimo de *w (m + w_min)*. Com isso podemos deslocar a função no eixo *y* do plano cartesiano, que não seria possível utilizando somente a fórmula original do co-seno. O comportamento desta função pode ser observado na Figura 1.

Com o valor do peso de inércia assumindo esse comportamento não-monotônico, a otimização seria da seguinte forma: quando o termo de inércia da PSO reduz, tendendo ao seu valor mínimo, as partículas perdem energia, se estabilizando em torno de um ponto de mínimo. Se este termo de inércia recebe valores maiores que 1, a tendência é a velocidade da partícula no instante anterior possuir maior importância, podendo causar um espalhamento do enxame. Se o termo de inércia atinge novamente valores menores, a tendência será o enxame se aproximar novamente, efetuando uma busca em torno de um mínimo local. Este mínimo, não necessariamente será o mesmo do primeiro. Portanto, o espalhamento do enxame pode ajudar o processo da PSO a fugir de mínimos locais, pois propicia uma busca global em um momento que seria menos provável encontrar outra melhor solução.

Com isso, a inércia *w* poderá atuar na diversificação e intensificação da PSO. O número de oscilações da função que controlará a inércia ao longo da PSO seria o número de tentativas de fugas de mínimos locais, já que a cada vez que *w* ultrapassar 1, o enxame poderá se espalhar.

Para melhor ilustrar o efeito desse ciclo não-monotônico do peso de inércia, os resultados experimentais com 4 funções de teste não-lineares são relatados e discutidos.

5. Configurações Experimentais

Para a comparação entre os dois modelos, foram utilizadas 4 funções não-lineares bem estudadas na literatura, sendo em todos os casos, problemas de minimização. As duas primeiras tratam-se de funções unimodais, e as duas últimas de funções multimodais. Todas apresentam seu mínimo global correspondente ao valor (0,0,...,0), de acordo com o seu número de coordenadas *n*.

A primeira função proposta é a função *Sphere*, descrita pela seguinte equação:

$$f_1(x) = \sum_{i=1}^{n} x_i^2$$

onde, como em todas as funções, x é um vetor n-dimensional de números reais, e x_i é o i-ésimo elemento deste vetor.

A segunda função é a função *De Jong F4 – no noise*, dada pela equação:

$$f_2(x) = \sum_{i=1}^{n} i \,.\, x_i^4$$

A terceira é a função *Rastrigin*, descrita pela equação:

$$f_3(x) = \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i) + 10)$$

A quarta e última função é a função *Griewank*, que é dada pela equação:

$$f_4(x) = \sum_{i=1}^{n} (x_i^2 / 4000) - \prod_{i=1}^{n} \cos(x_i / \sqrt{i}) + 1$$

A Tabela 1 lista o domínio de cada uma.

Tabela 1: Domínio da função

Função	Domínio
f_1	$-5.12 \leq x_i \leq 5.12$
f_2	$-20 \leq x_i \leq 20$
f_3	$-5.12 \leq x_i \leq 5.12$
f_4	$-600 \leq x_i \leq 600$

Para ambos os modelos de PSO, o número de iterações para cada função foi fixado em 2000, para cada uma das cinco dimensões testadas, correspondendo a 10, 20, 40, 60 e 80 coordenadas. O tamanho da população em cada execução foi estabelecido em 1000, sendo esta gerada a partir de uma distribuição uniforme, dentro dos intervalos especificados na Tabela 1. Os valores de c_1 e c_2 foram definidos iguais a 1,4. A vizinhança do tipo estrela foi utilizada.

Para a PSO padrão, conservando a idéia original do algoritmo, o valor da inércia *w* foi definido como uma subtração linear entre os valores 0,9 e 0,4.

Para a PSO com o controle de inércia *w*, teremos os seguintes parâmetros: o valor superior de *w* foi fixado em 1,2, o valor inferior de *w* foi 0,4 e os ciclos não-monotônicos são concluídos a cada 320 iterações. Todos esses valores da PSO com

controle de inércia foram obtidos através de um processo de ajuste paramétrico com base em testes estatísticos sobre a função Rastrigin, onde foi feita a análise sobre os resultados obtidos. Esse processo é semelhante ao descrito em [Oliveira *et al.* 2006].

Uma característica observada durante o estabelecimento de tais parâmetros para a PSO com o controle de inércia é em relação ao número de ciclos não-monotônicos ao longo da otimização. A utilização de muitas iterações para completar um ciclo não-monotônico não resulta em um bom desempenho do algoritmo, pois a otimização fica muito demorada devido ao elevado número de iterações para o espalhamento das partículas. Já a utilização de poucas iterações para completar um ciclo não-monotônico também não resulta em um bom desempenho do algoritmo, devido à rápida convergência do termo de inércia, fazendo com que o processo de otimização não se torne tão diversificado como na PSO padrão.

A Figura 1 mostra quais os valores a inércia *w* irá assumir ao longo da execução do algoritmo para as configurações apresentadas, tanto para o modelo padrão, com um decrescimento linear dessa inércia, quanto para o modelo com o controle da inércia da partícula, com as oscilações não-monotônicas.

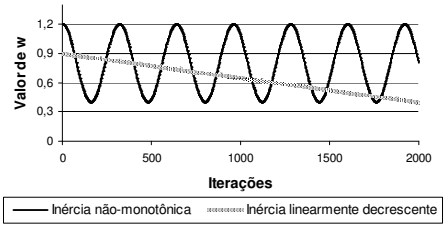

Figura 1: Valor da inércia *w* da partícula ao longo da execução do algoritmo

Um total de 30 testes em ambos os modelos, para cada configuração experimental, foi conduzido.

6. Resultados Finais e Discussões

Nesta seção, comparamos os resultados obtidos da PSO padrão e a PSO com o controle de inércia *w*, explorando os efeitos do espalhamento das partículas sobre a otimização.

A Tabela 2 mostra o resultado obtido na execução dos dois modelos da PSO, o padrão e o com controle de inércia. Ela lista a função utilizada para o espaço de busca, e, para sua dimensão, os respectivos valores médios da melhor partícula encontrada durante o processo de otimização, além do teste estatístico utilizado para a comparação dos resultados, mostrando também se houve diferença significativa entre os mesmos.

Para a análise dos resultados, foi, inicialmente, aplicado o teste de *Shapiro-Wilks* para verificar a normalidade dos dados. Quando a distribuição normal foi aceita, o teste *t de Student* foi aplicado (teste paramétrico). Já nas situações em que a distribuição normal não pôde ser aceita, foi aplicado o teste *U de Mann-Whitney* (teste não-paramétrico). As diferenças entre os resultados foram consideradas estatisticamente significantes quando o valor de "*p*" (*p-value*) foi menor que 0,05 (95% de confiança).

Já as Figuras 2 e 3, são os gráficos que ilustram alguns experimentos mostrados na Tabela 2, mostrando o desenvolvimento da otimização, para cada função utilizada com dimensão 60. No eixo y do gráfico temos a avaliação da partícula, enquanto que no

eixo x temos as iterações do processo de otimização. A Figura 3 é a mesma otimização que a Figura 2 (c), porém com uma escala maior.

Tabela 2: Média da avaliação da melhor partícula

Função	Dimensões	PSO padrão	PSO com controle de inércia	Teste estatístico	Diferença entre os resultados
f_1	10	0.000×10^{0}	0.000×10^{0}	t de Student	Insignificante
	20	0.000×10^{0}	0.000×10^{0}	t de Student	Insignificante
	40	0.000×10^{0}	4.807×10^{-5}	U de Mann-Whitney	Significante
	60	6.991×10^{0}	8.882×10^{-1}	U de Mann-Whitney	Significante
	80	2.447×10^{1}	6.421×10^{0}	U de Mann-Whitney	Insignificante
f_2	10	0.000×10^{0}	0.000×10^{0}	t de Student	Insignificante
	20	0.000×10^{0}	0.000×10^{0}	t de Student	Insignificante
	40	5.333×10^{3}	9.575×10^{-6}	U de Mann-Whitney	Significante
	60	1.067×10^{5}	3.200×10^{4}	U de Mann-Whitney	Significante
	80	6.667×10^{5}	2.988×10^{5}	U de Mann-Whitney	Insignificante
f_3	10	0.000×10^{0}	8.623×10^{-1}	U de Mann-Whitney	Significante
	20	7.796×10^{0}	7.428×10^{0}	U de Mann-Whitney	Insignificante
	40	9.366×10^{1}	7.263×10^{1}	U de Mann-Whitney	Significante
	60	2.276×10^{2}	2.001×10^{2}	t de Student	Significante
	80	3.866×10^{2}	3.024×10^{2}	t de Student	Significante
f_4	10	2.575×10^{-2}	5.208×10^{-2}	t de Student	Significante
	20	2.305×10^{-2}	2.304×10^{-2}	U de Mann-Whitney	Insignificante
	40	7.957×10^{-3}	6.278×10^{-2}	U de Mann-Whitney	Significante
	60	1.505×10^{1}	4.961×10^{-1}	U de Mann-Whitney	Significante
	80	7.528×10^{1}	2.263×10^{1}	U de Mann-Whitney	Insignificante

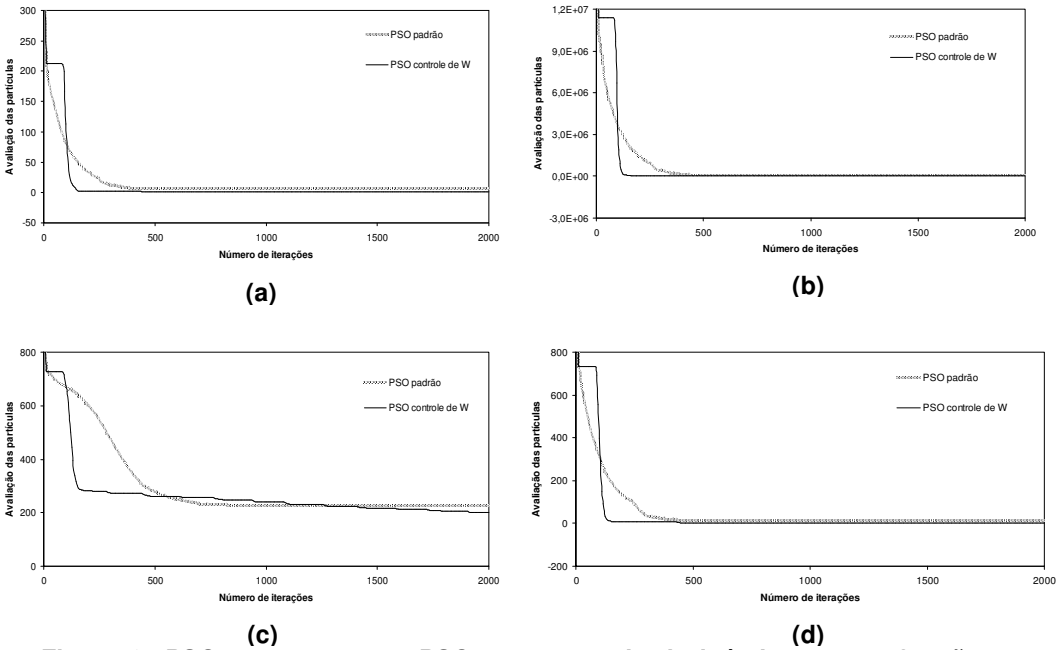

Figura 2: PSO padrao versus PSO com controle de inércia para as funções Sphere (a), De Jong F4 – no noise (b), Rastrigin (c) e Griewank (d)

Um ponto observado pelas Figuras 2 e 3 é a velocidade de convergência e a capacidade de encontrar uma boa solução da PSO com controle de inércia não-monotônico. Como o termo de inércia chega ao seu ponto de mínimo por volta da iteração 160, a PSO com o controle de inércia converge rapidamente para uma boa solução nessa iteração, em todos os casos. Para problemas simples, a melhor solução já

seria encontrada, dispensando assim, o espalhamento das partículas. Para problemas complexos, encontra certamente um mínimo local. Nesse ponto, o espalhamento das partículas seria interessante para tentar encontrar uma solução mais promissora.

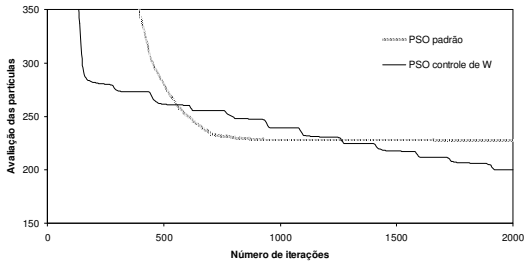

Figura 3: PSO padrão versus PSO com controle de inércia para a função Rastrigin em escala ampliada

Uma característica sensível do controle de inércia não-monotônico é sua capacidade de refinamento, que é bem compreendido observando-se a Figura 3. O controle de inércia não-monotônico, como visto, tem uma rápida capacidade de encontrar uma boa solução, mas sua capacidade de refinamento é mais lenta que a PSO padrão. Pela Figura 3, nota-se que durante todas as iterações, a PSO com controle de inércia ainda tentava buscar um melhor ponto de estabilização, enquanto que a PSO padrão já havia se estabilizado por volta da iteração 1000. Essa característica deve ser considerada, pois na maioria dos problemas, a estabilização da PSO com controle de inércia é mais demorada, principalmente para problemas mais complexos. Uma explicação para esse fato seria que a partir do momento do espalhamento das partículas até que a inércia volte a assumir valores menores que 1,0, em tais iterações, não haverá a busca por melhores soluções, mas o afastamento do melhor ponto já encontrado. Isso gera certa demora para encontrar a solução desejada.

Todavia, pelos resultados observados, ambos os métodos nos darão um bom resultado durante a otimização, mas, através do controle de inércia não-monotônico da partícula, teremos uma probabilidade maior de fuga de um mínimo local encontrado, devido a capacidade de busca global gerada pela energia inserida no sistema pelo controle de inércia, podendo assim levar a otimização a resultados mais satisfatórios.

7. Conclusões e trabalhos futuros

Neste trabalho, um modo diferente de trabalhar com a Otimização por Enxame de Partículas foi apresentado. Trata-se de uma forma de controlar a inércia da partícula de modo a permitir, muitas vezes, a fuga de soluções ótimas locais. Tal controle foi baseado em uma função não-monotônica, utilizando para isso a função co-seno, o que difere do modelo original, que é linearmente decrescente. Para os experimentos, foram utilizadas 4 funções para o espaço de busca bastante conhecidas na literatura.

Pelos testes, assim como em [Eberhart e Shi 1998], vimos que algumas vezes a PSO necessita de uma capacidade de busca global próximo ao fim do experimento, principalmente para problemas complexos, pelo fato de usarmos um termo de inércia decaindo linearmente. Assim, utilizando um termo de inércia não-monotônico, teremos essa capacidade de busca global várias vezes ao longo do processo de otimização, que, como visto, poderá trazer benefícios, principalmente para problemas complexos.

Para trabalhos futuros, a meta será desenvolver e testar novas funções para o controle da inércia, a fim de verificar o desempenho da otimização sobre as funções não-lineares do espaço de busca. Novas pesquisas deverão também investigar formas de melhorar a capacidade de refinamento da solução da PSO com o controle de inércia, que, como foi mostrado, ainda apresenta um menor desempenho em relação à PSO padrão, podendo levar a otimização a resultados inferiores. Além disso, outros valores dos parâmetros da PSO, como c_1, c_2 e dimensão do problema, deverão ser investigados para se tentar obter uma melhor otimização desses problemas.

Referências

Coelho, C.A., Toscano, G. e Lechuga, M.S. (2004), Handling Multiple Objectives with Particle Swarm Optimization. IEEE Transactions on Evol. Computation, Vol. 8, n° 3.

Shi, Y. e Eberhart, R. C. (1999) "Empirical Study of Particle Swarm Optimization", Proceedings of the 1999 Congress of Evolutionary Computation, vol. 3, 1945–1950.

Kennedy, J. e Eberhart, R. C. (1995), Particle swarm optimization. Proc. IEEE International Conference on Neural Networks (Perth, Australia), IEEE Service Center, Piscataway, NJ, pp. IV: 1942-1948.

Kennedy, J., Eberhart, R. e Shi, Y. (2001), Swarm Intelligence, Morgan Kaufmann Publishers. Inc, San Francisco, CA, 2001.

Eberhart, R. C. e Shi, Y. H. (1998), Comparison between genetic algorithms e particle swarm optimization. Annual Conference on Evolutionary Programming, San Diego.

Angeline, P. J. (1998), Using selection to improve particle swarm optimization. IEEE International Conference on Evolutionary Computation, Anchorage, Alaska, May 4-9.

Shi, Y. e Eberhart, R. C. (1998) "Parameter Selection in Particle Swarm Optimization", Evolutionary Programming VII, Lecture Notes in Computer Science 1447, 591–600.

Suganthan, P. N. (1999) "Particle Swarm Optimiser with Neighbourhood Operator", Proceedings of the 1999 Congress of Evolutionary Computation, vol. 3, 1958–1962.

Kennedy, J. (1999) "Small Worlds e Mega-Minds: Effects of Neighborhood Topology on Particle Swarm Performance", Proceedings of the 1999 Congress of Evolutionary Computation, vol. 3, 1931–1938. IEEE Press.

Lvbjerg, M., Rasmussen, T. K. e Krink, T. (2001) "Hybrid Particle Swarm Optimiser with Breeding e Subpopulations", Proceedings of the Genetic e Evolutionary Computation Conference.

Oliveira, H. C. B., Vasconcelos, G. C., Mesquita, R. V. e Souza, M. M. (2006), Parametrical Adjusting of a Hybrid System for the Vehicle Routing Problem with Time Windows. HIS'06, Auckland, New Zealand. IEEE Computer Society.

Jiao, B., Lian, Z. e Gu, X. (2008) "A dynamic inertia weight particle swarm optimization algorithm" Chaos, Solitons & Fractals - vol. 37, Issue 3, August 2008, Pages 698-705

Yang, X., Yuan, J., Yuan, J. e Mao H. (2007) "A modified particle swarm optimizer with dynamic adaptation", Applied Mathematics and Computation, vol. 189, Issue 2, June 2007, Pages 1205-1213